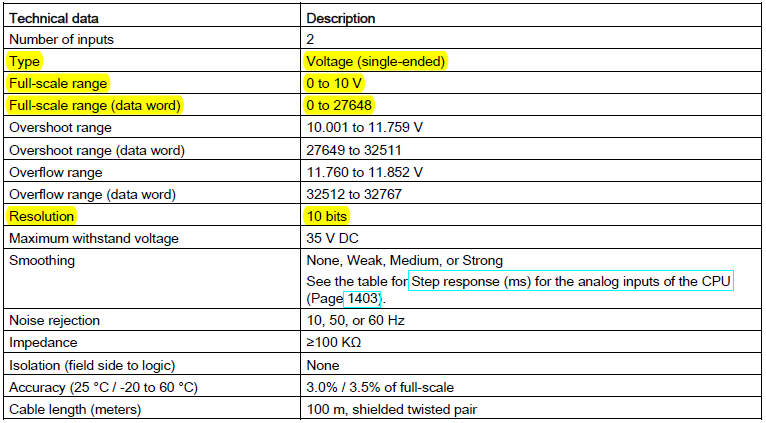
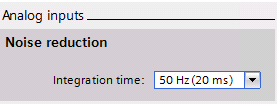
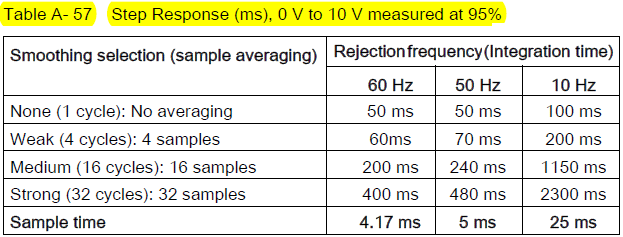
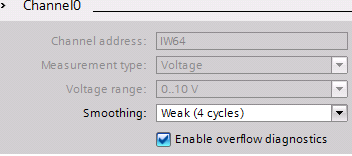
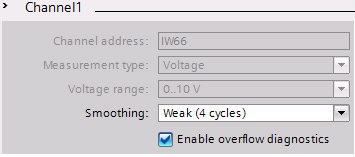
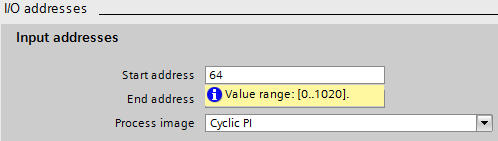
**Analog Girişler**

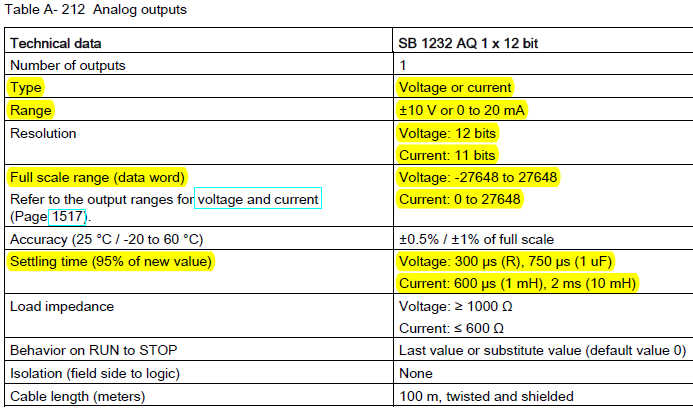


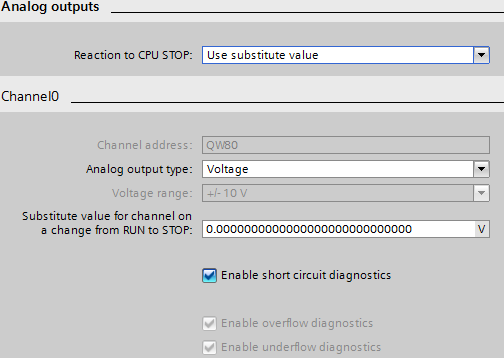
 

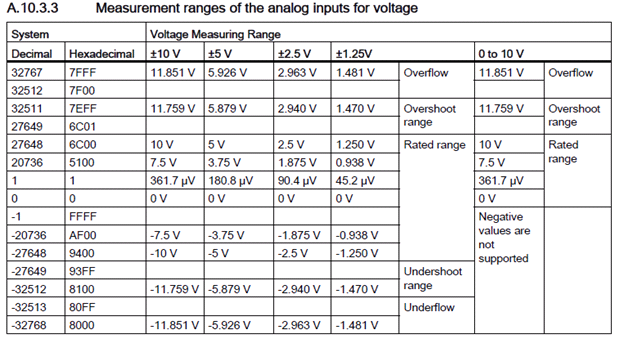
 

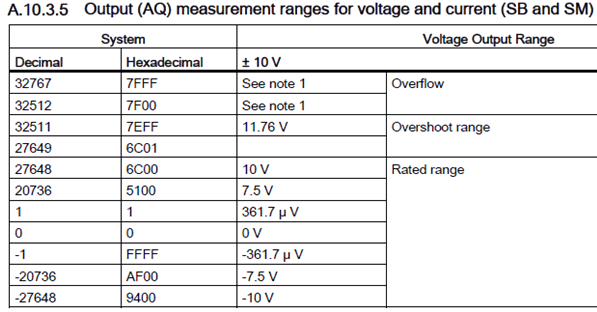


**Analog Çıkışlar**

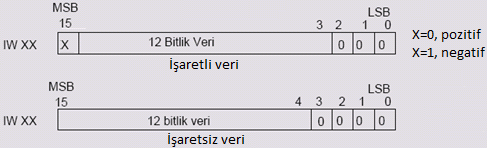






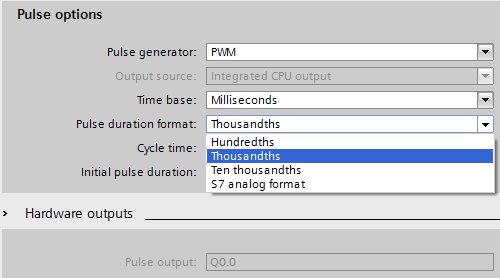
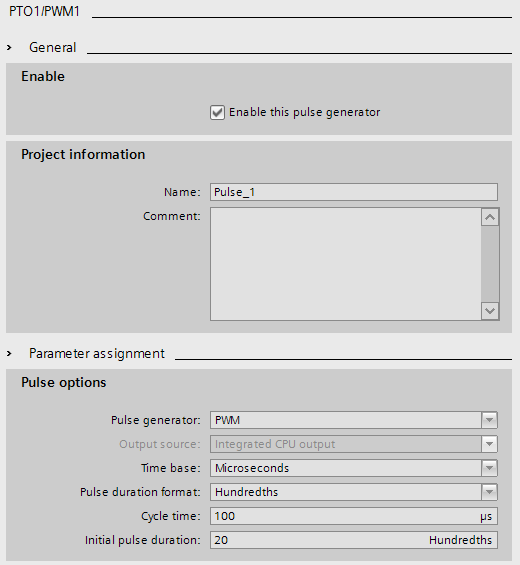


**S7 – Analog Format:**

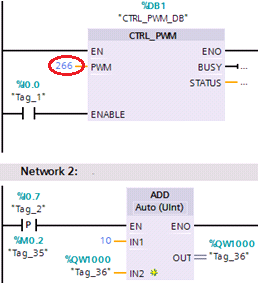


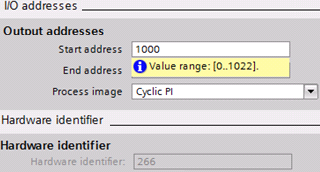
**4 adet PTO / PWM**

PTO (Pulse Train Output); %50 Duty, T değişken, Servo-Step Motor Uygulamaları

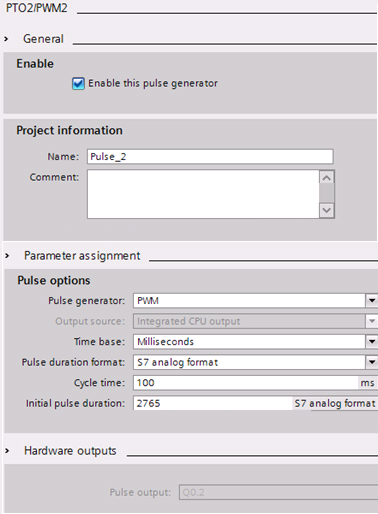
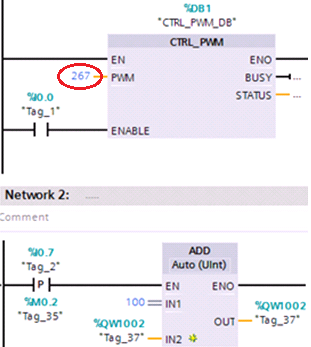


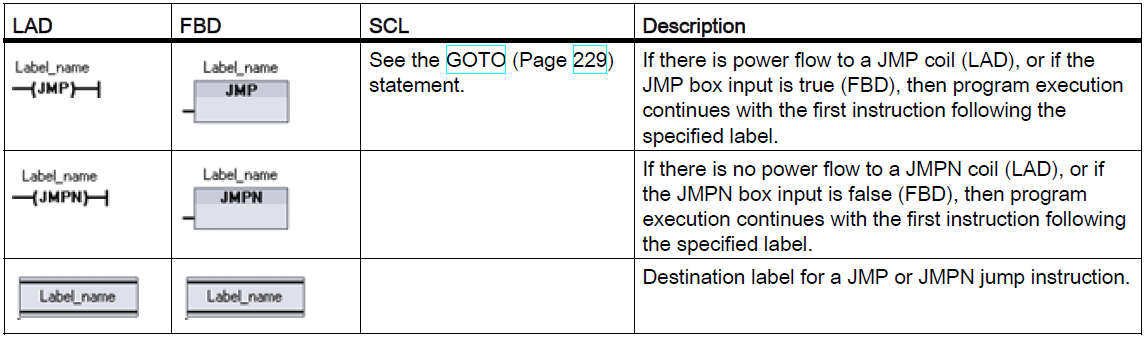






**Donanım ayarları değiştirildiğinde yazılıma ilave olarak donanım da yeniden derlenmelidir.**

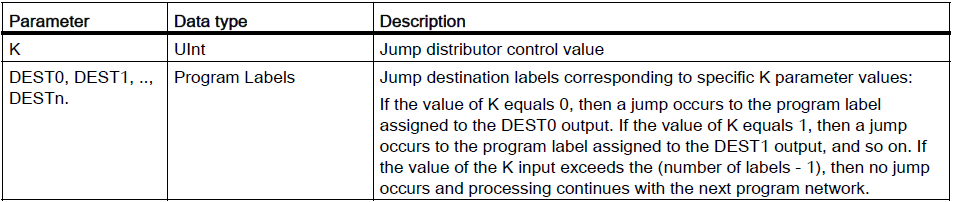
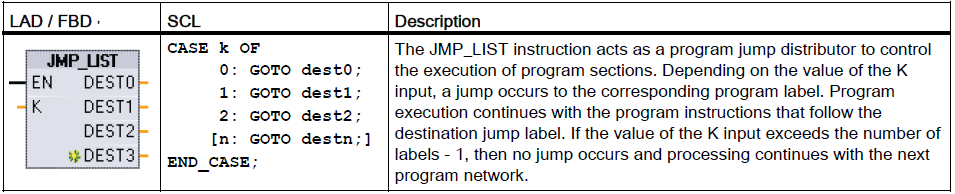
**Program Akış Kontrol Komutları**

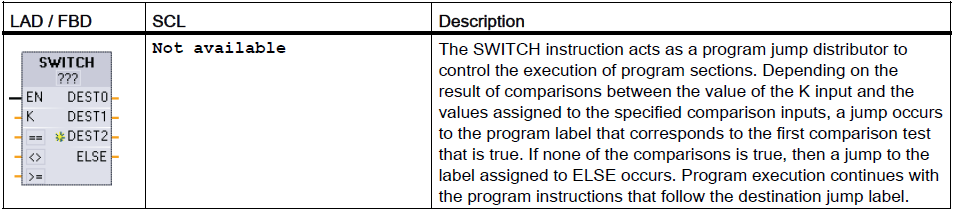
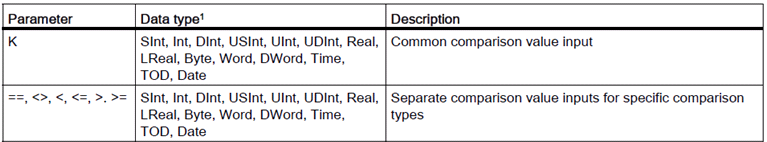
● You can jump forward or backward.

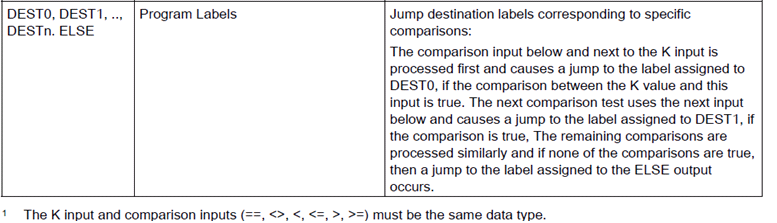
● Each label must be unique within a code block.

● You can jump to the same label from more than one place in the same code block.

● You can jump within a code block, but you cannot jump from one code block to another code block.

**JMP\_LIST instruction**

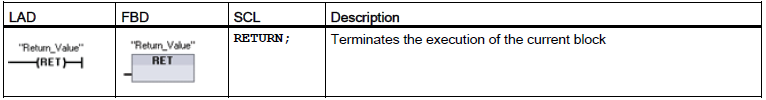
**SWITCH instruction** 



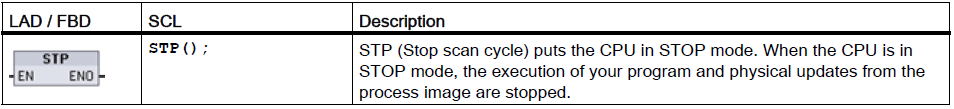
**RET execution control instruction**

The optional RET instruction is used to terminate the execution of the current block. If and only if there is power flow to the RET coil (LAD) or if the RET box input is true (FBD), then program execution of the current block will end at that point and instructions beyond the RET instruction will not be executed. If the current block is an OB, the "Return\_Value" parameter is ignored. If the current block is a FC or FB, the value of the "Return\_Value " parameter is passed back to the calling routine as the ENO value of the called box.

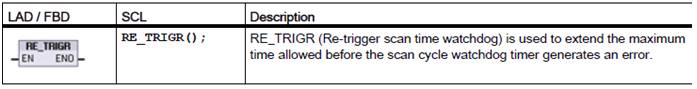
You are not required to use a RET instruction as the last instruction in a block; this is done automatically for you. You can have multiple RET instructions within a single block.



**Stop scan cycle instruction**



**Re-trigger scan cycle watchdog instruction**



Use the RE\_TRIGR instruction to restart the scan cycle monitoring timer during a single scan cycle. This has the effect of extending the allowed maximum scan cycle time by one maximum cycle time period, from the last execution of the RE\_TRIGR function.

**Note**

Prior to S7-1200 CPU firmware version 2.2, RE\_TRIGR was restricted to execution from a program cycle OB and could be used to extend the PLC scan time indefinitely. ENO = FALSE and the watchdog timer is not reset when RE\_TRIGR was executed from a start up OB, an interrupt OB, or an error OB.

For firmware version 2.2 and later, RE\_TRIGR can be executed from any OB (including start up, interrupt, and error OBs). However, the PLC scan can only be extended by a maximum of 10x the configured maximum cycle time.

Setting the PLC maximum cycle time Configure the value for maximum scan cycle time in the Device configuration for "Cycle time".



**Watchdog timeout**

If the maximum scan cycle timer expires before the scan cycle has been completed, an error is generated. If an error handling code block OB 80 is included in the user program, the CPU executes OB 80 where you may add program logic to create a special reaction. If OB 80 is not included, the first timeout condition is ignored and the CPU goes to STOP.

If a second maximum scan time timeout occurs in the same program scan (2 times the maximum cycle time value), an error is triggered that causes the CPU to transition to STOP mode.

In STOP mode, your program execution stops while CPU system communications and system diagnostics continue

**System and clock memory**

You use the CPU properties to enable bytes for "system memory" and "clock memory". Your program logic can reference the individual bits of these functions ***by their tag names***.

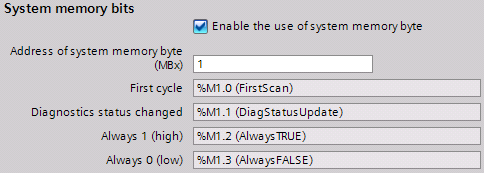
You can assign one byte in M memory for system memory. The byte of system memory provides the following four bits that can be referenced by your user program by the following tag names:

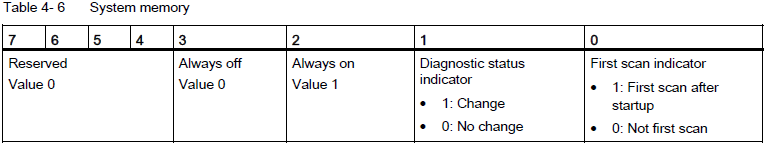
– ***First cycle: (Tag name "FirstScan")*** bit is set to1 for the duration of the first scan after the startup OB finishes. (After the execution of the first scan, the "first scan" bit is set to 0.)

– ***Diagnostics status changed (Tag name: "DiagStatusUpdate")*** is set to 1 for one scan after the CPU logs a diagnostic event. Because the CPU does not set the "diagnostic graph changed" bit until the end of the first execution of the program cycle OBs, your user program cannot detect if there has been a diagnostic change either during the execution of the startup OBs or the first execution of the program cycle OBs.

– ***Always 1 (high): (Tag name "AlwaysTRUE")*** bit is always set to 1.

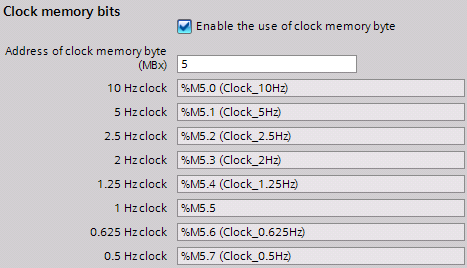
– ***Always 0 (low): (Tag name "AlwaysFALSE")*** bit is always set to 0.





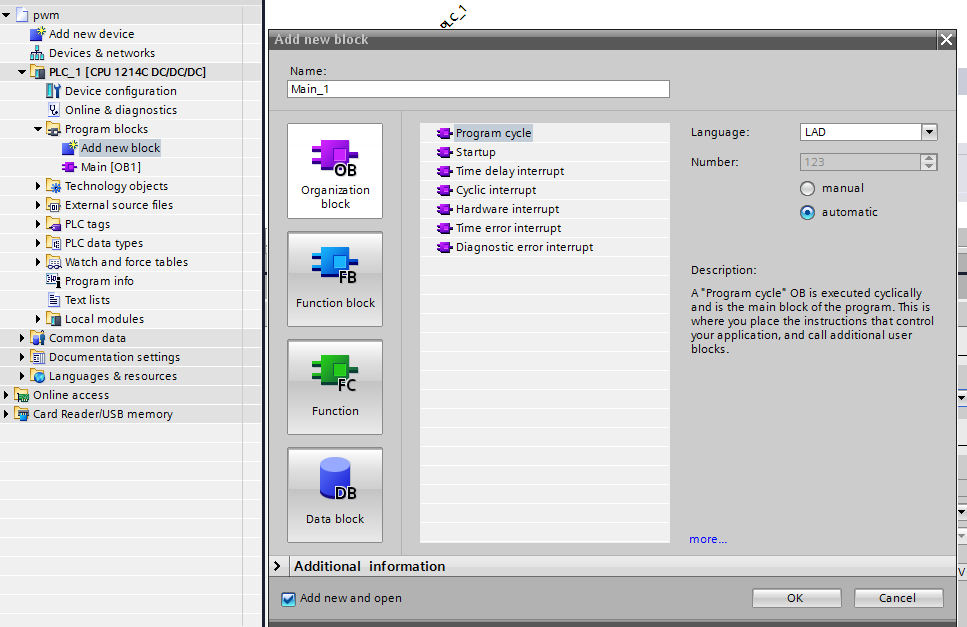
You can assign one byte in M memory for **clock memory**. Each bit of the byte configured as clock memory generates a square wave pulse. The byte of clock memory provides 8 different frequencies, from 0.5 Hz (slow) to 10 Hz (fast). You can use these bits as control bits, especially when combined with edge instructions, to trigger actions in the user program on a cyclic basis.

The CPU initializes these bytes on the transition from STOP mode to STARTUP mode. The bits of the clock memory change synchronously to the CPU clock throughout the STARTUP and RUN modes.



**Kesmeler (INTERRUPTs)**

Interrupts can occur during any part of the scan cycle, and are event-driven. When an event occurs, the CPU interrupts the scan cycle and calls the OB that was configured to process that event. After the OB finishes processing the event, the CPU resumes execution of the user program at the point of interruption.



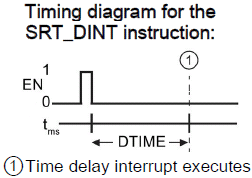
**1- Time delay interrupt OB:** Time delay interrupt OBs execute after a time delay that you configure. You configure time delay interrupt events to occur after a specified delay time has expired. You assign the delay time with the SRT\_DINT instruction. The time delay events interrupt the program cycle to execute the corresponding time delay interrupt OB. You can attach only one time delay interrupt OB to a time delay event. The CPU supports **four time delay events**.

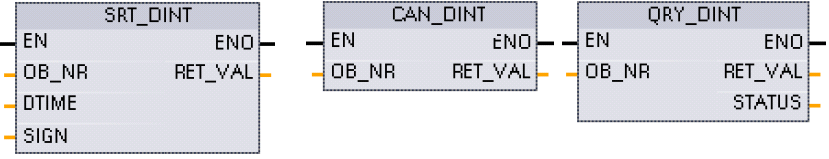
You can **start and cancel time delay interrupt processing with the SRT\_DINT and CAN\_DINT instructions**, or query the interrupt status with the QRY\_DINT instruction. Each time delay interrupt is a one-time event that occurs after the specified delay time. **If the time delay event is cancelled before the time delay expires, the program interrupt does not occur**. When EN=1, the SRT\_DINT instruction starts the internal time delay timer (DTIME). When the time delay elapses, the CPU generates a program interrupt that triggers the execution of the associated time delay interrupt OB. You can cancel an in-process time delay interrupt before the specified time delay occurs by executing the CAN\_DINT instruction. The total number of active time delay interrupt events **must not exceed four**.

You can only assign time delay interrupt OBs to the SRT\_DINT and CAN\_DINT instructions. No time delay interrupt OB exists in a new project. **You must add time delay interrupt OBs to your project.**

The SRT\_DINT starts the time delay timer on every scan when EN=1. **Assert EN=1 as a**

**one-shot rather than just setting EN=1 to begin your time delay.**





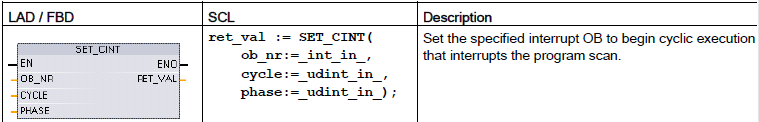
**2- Cyclic interrupt OB:** Cyclic interrupt OBs execute at a **specified time interval**. You can configure up to a **total of four cyclic interrupt events, with one OB corresponding to each cyclic interrupt event. You can attach only one cyclic interrupt OB to a cyclic event.**

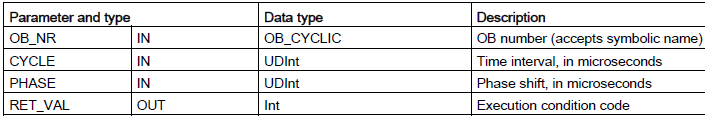
The cyclic interrupt events allow you to configure the execution of an interrupt OB at a configured cycle time. You configure the initial cycle time when you create the cyclic interrupt OB. A cyclic event interrupts the program cycle and executes the corresponding cyclic

interrupt OB. Note that the cyclic interrupt event is at a higher priority class than the program cycle event.

You can assign a phase shift to each cyclic interrupt so that the execution of cyclic interrupts can be offset from one another by the phase offset amount. For example, if you have a 5 ms cyclic event and a 10 ms cyclic event, every 10 ms both events occur at the same moment. If you phase shift the 5 ms event by 1 to 4 ms and the 10 ms event by 0 ms, then the two events do not occur at the same moment. The default phase offset is 0. The maximum phase offset is 6000 ms (6 seconds) or the maximum Cyclic time, whichever is smaller.

You can also query and change the scan time and the phase shift from your program using the Query cyclic interrupt (QRY\_CINT) and Set cyclic interrupt (SET\_CINT) instructions. Scan time and phase shift values set by the SET\_CINT instruction do not persist through a power cycle or a transition to STOP mode; scan time and phase shift values return to the initial values following a power cycle or a transition to STOP. The CPU supports a total of four cyclic interrupt events.





Examples: time parameter

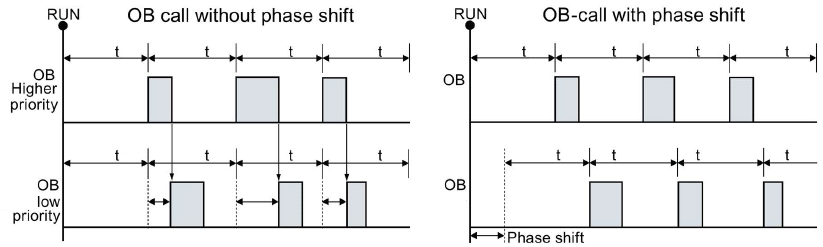
● If the CYCLE time = 100 us, then the interrupt OB referenced by OB\_NR interrupts the cyclic program scan every 100 us. The interrupt OB executes and then returns execution control to the program scan, at the point of interruption.

● If the CYCLE time = 0, then the interrupt event is deactivated and the interrupt OB is not executed.

● The PHASE (phase shift) time is a specified delay time that occurs before the CYCLE time interval begins. You can use the phase shift to control the execution timing of lower priority OBs.

If lower and higher priority OBs are called in the same time interval, the lower priority OB is only called after the higher priority OB has finished processing. The execution start time for the low priority OB can shift depending on the processing time of higher priority OBs.

If you want to start the execution of a lower priority OB on a fixed time cycle, then phase shift time should be greater then the processing time of higher priority OBs.



**3- Hardware interrupt OB**: Hardware interrupt OBs execute when the relevant hardware event occurs. A hardware interrupt OB interrupts normal cyclic program execution in reaction to a signal from a hardware event.

Changes in the hardware, such as a rising or falling edge on an input point, or an HSC event trigger hardware interrupt events. The S7-1200 supports one interrupt OB for each hardware interrupt event. You enable the hardware events in the device configuration, and assign an OB for an event in the device configuration or with an ATTACH instruction in the user program. The CPU supports several hardware interrupt events. The CPU model and the number of input points determine the exact events that are available.

**4- Time error interrupt OB:** If configured, the time error interrupt OB (OB 80) executes when either the scan cycle exceeds the maximum cycle time or a time error event occurs. If triggered, it executes, interrupting normal cyclic program execution or any other event OB. The occurrence of either of these events generates a diagnostic buffer entry describing the event. The diagnostic buffer entry is generated regardless of the existence of the time error interrupt OB.

The occurrence of any of several different time error conditions results in a time error event:

● Scan cycle exceeds maximum cycle time

The "maximum cycle time exceeded" condition results if the program cycle does not complete within the specified maximum scan cycle time. See the section on "Monitoring the cycle time in the S7-1200 Manual" (Page 109) for more information regarding the maximum cycle time condition, how to configure the maximum scan cycle time in the properties of the CPU, and how to reset the cycle timer.

● CPU cannot start requested OB because a second time interrupt (cyclic or time-delay) starts before the CPU finishes execution of the first interrupt OB

● Queue overflow occurred The "queue overflow occurred" condition results if the interrupts are occurring faster than the CPU can process them. The CPU limits the number of pending (queued) events by using a different queue for each event type. If an event occurs when the corresponding queue is full, the CPU generates a time error event. All time error events trigger the execution of the time error interrupt OB if it exists. If the time error interrupt OB does not exist, then the device configuration of the CPU determines the CPU reaction to the time error:

● The default configuration for time errors, such as starting a second cyclic interrupt before the CPU has finished the execution of the first, is for the CPU to stay in RUN.

● The default configuration for exceeding the maximum time is for the CPU to change to STOP.

The user program can extend the program cycle execution time up to ten times the configured maximum cycle time by executing the RE\_TRIGR instruction (Page 309) to restart the cycle time monitor. However, if two "maximum cycle time exceeded" conditions occur within the same program cycle without resetting the cycle timer, then the CPU transitions to STOP, regardless of whether the time error interrupt OB exists

Hardware interrupt events

The following hardware interrupt events are supported by the CPU:

● Rising edge events (first 12 built-in CPU digital inputs (DIa.0 to DIb.3) and all SB digital inputs)

● Falling edge events (first 12 built-in CPU digital inputs (DIa.0 to DIb.3) and all SB digital inputs)

● High-speed counter (HSC) current value = reference value (CV = RV) events (HSC 1 through 6)

● HSC direction changed events (HSC 1 through 6)

– A direction changed event occurs when the HSC is detected to change from increasing to decreasing, or from decreasing to increasing.

● HSC external reset events (HSC 1 through 6)

– Certain HSC modes allow the assignment of a digital input as an external reset that is used to reset the HSC count value to zero. An external reset event occurs for such a HSC, when this input transitions from OFF to ON.

**Enabling hardware interrupt events in the device configuration**

Hardware interrupts **must be enabled during the device configuration**. You must check the enable-event box in the device configuration for a digital input channel or a HSC, if you want to attach this event during configuration or run time.

Check box options within the PLC device configuration:

● Digital input,

– Enable rising edge detection

– Enable falling edge detection

● High-speed counter (HSC)

– Enable this high-speed counter for use

– Generate interrupt for counter value equals reference value count

– Generate interrupt for external reset event

– Generate interrupt for direction change event

**Adding new HARDWARE interrupt OB code blocks to your program**

By default, no OB is attached to an event when the event is first enabled. This is indicated by the "HW interrupt:" device configuration "<not connected>" label. Only hardware-interrupt OBs can be attached to a hardware interrupt event. All existing hardware-interrupt OBs appear in the "HW interrupt:" drop-down list. If no OB is listed, then you must create an OB of type "Hardware interrupt" as follows. Under the project tree "Program blocks" branch:

1. Double-click "Add new block", select "Organization block (OB)" and choose "Hardware interrupt".

2. Optionally, you can rename the OB, select the programming language (LAD or FBD), and select the block number (switch to manual and choose a different block number than that suggested).

3. Edit the OB and add the programmed reaction that you want to execute when the event occurs. You can call FCs and FBs from this OB, to a nesting depth of four.

**OB\_NR parameter**

All existing hardware-interrupt OB names appear in the device configuration "HW interrupt:" “drop-down” list and in the ATTACH / DETACH parameter OB\_NR drop-list.

**EVENT parameter**

When a hardware interrupt event is enabled, a unique default event name is assigned to this particular event. You can change this event name by editing the "Event name:" edit box, but it must be a unique name. These event names become tag names in the "Constants" tag table, and appear on the EVENT parameter drop-down list for the ATTACH and DETACH instruction boxes. The value of the tag is an internal number used to identify the event.

**General operation**

***Each hardware event can be attached to a hardware-interrupt OB which will be queued for execution when the hardware interrupt event occurs. The OB-event attachment can occur at configuration time or at run time.***

You have the option to **attach** or **detach** an OB to an enabled event at configuration time. To attach an OB to an event at configuration time, you must use the "HW interrupt:" drop-down list (click on the down arrow on the right) and select an OB from the list of available hardware-interrupt OBs. Select the appropriate OB name from this list, or select "<not connected>" to remove the attachment.

You can also attach or detach an enabled hardware interrupt event during run time. Use the ATTACH or DETACH program instructions during run time (multiple times if you wish) to attach or detach an enabled interrupt event to the appropriate OB. If no OB is currently attached (either from a "<not connected>" selection in device configuration, or as a result of executing a DETACH instruction), the enabled hardware interrupt event is ignored.

DETACH operation

Use the DETACH instruction to detach either a particular event or all events from a particular OB. If an EVENT is specified, then only this one event is detached from the specified OB\_NR; any other events currently attached to this OB\_NR will remain attached. If no EVENT is specified, then all events currently attached to OB\_NR will be detached.

**5- Diagnostic error interrupt OB:** The diagnostic error interrupt OB executes when the CPU detects a diagnostic error, or if a diagnostics-capable module recognizes an error and you have enabled the diagnostic error interrupt for the module. The diagnostic error interrupt OB interrupts the normal cyclic program execution. You can include an STP instruction in the diagnostic error interrupt OB to put the CPU in STOP mode if you desire your CPU to enter STOP mode upon receiving this type of error.

If you do not include a diagnostic error interrupt OB in your program, the CPU ignores the error and stays in **RUN** mode.

**Attach and detach instructions**

You can activate and deactivate interrupt event-driven subprograms with the ATTACH and DETACH instructions.

